CS1132 Fall 2012 Assignment 1a

> Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 3 of this document.

Do not use the `break` or `continue` statement in any homework or test in CS1132.

1 Playing Cards

1.1 Drawing one card from a deck

A standard deck of playing cards has 52 cards: there are four suits (Clubs, Diamonds, Hearts, and Spades) and each suit includes cards with the ranks A, 2, 3, …, 9, 10, J, Q, K. (There is no Joker in our deck of cards.) Implement the following function:

```
function [rk, st] = OneCard()
% Draw one card at random from a standard deck of 52 cards.
% rk represents the rank of the card: 1, 2, ..., or 13 where 1 indicates
%   Ace and 11, 12, and 13 indicate Jack, Queen, and King, respectively.
% st represents the suit of the card: 1, 2, 3, or 4, indicating Clubs,
%   Diamonds, Hearts, and Spades, respectively.
```

1.2 ∞-card shuffler

Write a script **`InfShuffler`** to display the first 20 cards issued from a shuffle machine that holds 20 decks of cards. Since 20 decks are shuffled in the machine and you need to display 20 cards, it is possible for cards of the same suit and rank to appear multiple times. Hint: your simulation for this problem can use a simple algorithm—it is not necessary to check for duplicates.

Make effective use of function **`OneCard`** from problem 1.1. Display each card by printing its suit and rank neatly, using the actual suit names and rank symbols (J instead of 11, A instead of 1, etc.). Below are several lines of example output:

```
No.      Card
 1      Q Hearts
 2      5 Hearts
 3      6 Clubs
 4      4 Clubs
 5      3 Clubs
 6      6 Clubs
 7      Q Spades
 8      7 Diamonds
 9      5 Spades
10      5 Clubs
11      J Diamonds
```

1.3 Counting red cards

Modify your script **`InfShuffler`** to display the sequence of cards issued from the 20-deck shuffle machine, stopping after 12 *red* cards (Hearts and Diamonds) have been issued or after 20 cards in total have been issued, whichever occurs first. Use the same print format as shown above in 1.2. Save this file as **`Sequence.m`**. (As in problem 1.2, there is no need to check for duplicate cards since you are printing no more than 20 cards from 20 decks.)

1.4 Game probabilities

You will use *simulation* to estimate the probabilities of some game events, making effective use of `OneCard` from problem 1.1. In this game, you draw one card from a standard deck. Let cases A, B, and C be defined as follows:

A: the drawn card has the rank J, Q, or K
B: the drawn card is red and has the rank 9 or 10
C: the drawn card is red

Write two functions to simulate *n* times the drawing of one card from a standard deck. (Note: each time you draw from a complete deck.) One function estimates the probability of $A \bigcup B$ and the other estimates the probability of $A \bigcap C$. The probability of an event is estimated as the number of occurrences of that event divided by the number of trials.

Implement the following two functions:

```
function p = AorB(n)
% p is the probability of case A or case B occurring when drawing one card
%   from a standard deck.
% n is the number of trials.

function p = AandC(n)
% p is the probability of both case A and case C occurring when drawing one
%   card from a standard deck.
% n is the number of trials.
```

2 Self-check list

The following is a list of the minimum necessary criteria that your assignment must meet in order to be considered satisfactory. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time. Although all these criteria are necessary, meeting all of them might still not be sufficient to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Comment your code! Make sure your function is properly commented, regarding function purpose and input/output parameters.
- Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Make sure your functions names are exactly the ones we have specified, including case.
- Check that the number and order of input and output parameters for each function match exactly the specifications we have given.
- Test each function independently, whenever possible, or write short scripts to test them.
- Check that your scripts and functions do not crash (i.e., end unexpectedly with an error message) or run into an infinite loop. Check your script several times in a row: before each test run, type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

3 Submission instructions

1. Upload files `OneCard.m`, `InfShuffler.m`, `Sequence`.m, `AorB`.m, and `AandC`.m to CMS in the submission area corresponding to Assignment 1a.
2. *After grading:* If you resubmit the assignment, upload your corrected files and be sure to select the **Regrade Request** option.